An agent based architecture for constructing Interactive Simulation Systems

Zhao, Z.

Publication date
2004

Chapter 7

Summary and discussion

7.1 Summary

In scientific research, interactive simulations play an increasingly important role; compared to conventional simulation systems, they allow parameter spaces to be explored more efficiently and computing processes to be controlled more accurately. In this way they also help to optimise the resource utilisation for both computing and storage, and to improve the efficiency of communication. However, the development and integration of the simulation and visualisation kernels is expensive. Traditional development methods often result in systems that have a limited adaptability due to the strong dependency between the system functionality and application specific logic control. The high costs and limited flexibility hamper the introduction of such systems. The available software architectures and middlewares for simulation systems mostly focus on the interoperability between system components, but not explicitly on the support for rapidly prototyping ISSs and for flexibly controlling the system behaviour.

In this thesis, we state that *the separation of application specific logic control from system functionality constitutes a crucial step in an improved development process for ISS*. We developed an agent-based component architecture, called Interactive Simulation System Conductor, implementing such a separation. It encapsulates the control intelligence for the integrated system as different roles of agents, and allows to employ these agents to realise a layered interconnection between components. The proof of concept implementation is based on High Level Architecture, a dedicated middleware for distributed simulation systems.

In the second chapter, we introduce the basic architecture of ISS-Conductor. Encapsulating the computing kernels and support structures of the principal modules of an ISS, such as simulation, visualisation and interaction, it allows scientists to assemble simulation experiments at a high level without concerning themselves with the low level details of integration. In the architecture, Communication Agents realise the basic interoperability between components; Module Agents orchestrate the

run-time system behaviour. The Run-Time Infrastructure of HLA is employed as the software bus for binding the distributed components.

In the third chapter, we discuss the functional design of ISS-Conductor. In an ISS-Conductor component, the MA controls the component behaviour using a reasoning kernel; it has knowledge about both the capabilities of the components and the application specific interaction constraints in its knowledge base. The component functionality is modelled as a finite state machine (*capability*), which can be programmed with the other components using a Petri net based mechanism (*scenario net*). At run time, MAs collaboratively interpret the interaction constraints and realise the control of the overall system behaviour.

The implementation details and performance characteristics of ISS-Conductor are discussed in chapter four. The ComA interfaces with the basic RTI services for data sharing and message passing. The reasoning kernel of the MA is realised using Prolog. We measure the performance of the implementation, and conclude that the Communication Agents add acceptable overheads to the RTI. ISS Conductor-based applications can reach a network utilisation comparable to that of pure TCP sockets for large size data objects. The logic control in the reasoning kernel also adds only a small overhead.

In the fifth chapter, we employ the ISS-Conductor architecture to prototype an interactive simulation environment for planning vascular operations. We discuss the detailed procedures for developing an ISS-Conductor component and for assembling the components into an interactive simulation system. In various test cases, we discuss the scenario level activity control, automatic performance tuning in a run-time system and collaborative interaction support. The experimental results show that ISS-Conductor only adds a small overhead to the legacy computing kernels.

In the sixth chapter, we discuss the feasibility of including ISS-Conductor compliant components as software resources in a Problem Solving Environment. One of the crucial issues we investigate is the automatic composition of an interactive simulation based experiment. ISS-Studio, a Java based environment has been prototyped in this chapter.

7.2 Conclusions and discussion

In the thesis, we describe the layered architecture of ISS-Conductor and discussed its implementation details. We demonstrated its capability to realise the separation between the ISS functionality and the application logic control, and therefore to fulfil the mission that we set out for in the beginning of this thesis: providing a layered framework for rapid prototyping of interactive simulation systems. A proof of concept implementation is based on HLA. In this final chapter of the thesis, we will not repeat

the conclusions drawn in the previous chapters, instead we discuss the architecture from the perspective of the underlying middleware.

The continuous growth of computing power and the design of novel middleware offer new possibilities for realising component integration and interoperability, but they also introduce incompatibility between legacy implementations and the new platforms. The heterogeneity of the computing platforms will be permanent. The solution used in the software industry is to establish a steering group, e.g. the OMG [212], from the main members of the community to formulate platform independent architectures for the development. However, the software development in academia can not always benefit from such mechanism. One of the reasons is that the effort in developing novel simulation and visualisation systems mostly lies on studying the domain problem rather than the engineering itself. The problem for integrating legacy simulation and visualisation systems is thus going to remain and there will be no single solution to all situations. In this thesis, we studied this problem from the perspective of rapidly prototyping ISSs, and contributed an agent based architecture to couple simulation and visualisation systems.

Considering a component based ISS as an analogue of an electronic circuit system, an ISS-Conductor component is more like an *intelligent card* than a *silicon chip*; it encapsulates the necessary intelligence for integrating a number of functional units together with the control for their run-time behaviour. There are a number of reasons for us to choose this vision. First, one of the missions for developing ISS-Conductor is to support the rapid prototyping of interactive simulation system. Using ISS-Conductor components, an ISS can be defined at the level of system interactions, which is easy to map onto the description of the domain problem. The assembly of small size units is considered to be a task for the component developer rather than for the research scientists. Second, this mechanism also has weak requirements on the underlying framework; ISS-Conductor only requires necessary services for distributing data objects and timestamp-ordered messages. The ComAs in the architecture localise the dependencies on the software bus and a component can be equipped with different implementations of the ComA to achieve the portability. Third, the run-time activities are controlled by the components autonomously instead of by a separate co-ordinator. It can have flexible paradigms for controlling the execution, both distributed and centralised.

HLA is a suitable architecture for distributed simulations, but it will not be the final solution. In this thesis, we did not discuss the issues on integrating the ISS-Conductor architecture with the Grid platforms. Actually, work along this line has been initiated in a European project, CrossGrid. In [213], we proposed a layered approach to realise the run-time binding between HLA compliant components on Grid environments. At the RTI level, the communication endpoint of the RTI execution is wrapped as a service, which can be discovered by an HLA application to create a federation. At the federation level, the basic services for managing HLA applications remain, only the data distribution between federates is handled using the Grid

enabled facilities, such as Grid FTP. At the federate level, the services originally provided by the RTI will be wrapped as Grid services. The contribution from this work can be applied to ISS-Conductor, since from the implementation point of view ISS-Conductor components are HLA compliant federates. The migration will not change the main design of ISS-Conductor but only the implementation of the ComAs. Further progress is discussed in [214, 215].

7.3 Future work

Based on the results of this thesis, a number of future research lines can be envisaged:

1. A lesson learned from VLAM-G project is that scientists will not choose a novel architecture simply because it looks beautiful unless it can work with the existing ones and provide exciting new features [216]. An important future work is to study the mapping schemes between existing architectures and ISS-Conductor so that legacy architectures can get benefit of the layered integration from ISS-Conductor in an easy and efficient way.

2. Looking at the underlying middleware, new integration paradigms provided in Grid environments, such as service-oriented integration, will be considered as new basis for ISS-Conductor.

3. The work described in chapter 6 is still on going. Including ISS components as software resources and sharing them among different organisations and projects is one of the core paradigms of combining ISSs and Grid environments. The research issues for semantic based discovery and intelligent planning will form another line of future work.